

## REMARKS

In response to the Office Action mailed on October 05, 2005, Applicants respectfully request reconsideration. Claim 1, 3-14, 16-26, and 27-42 are now pending in this Application. Claims 1, 15, and 26 are independent claims and the remaining claims are dependent claims. In this Amendment, claims 1, 15 and 26 have been amended, claims 2, 16 and 27 have been cancelled and claims 40-42 have been added. A version of the claims containing markings to show the changes made is included hereinabove. Applicants believe that the claims as presented are in condition for allowance. A notice to this affect is respectfully requested.

### Rejections under § 112

The Examiner rejected claims 1-39 under 35 U.S.C. §112, second paragraph, as being indefinite. Applicants respectfully disagree with the Examiner's assertion. Specifically, the Examiner stated that the claims recite "instructions" and "at least one function" and that it is not clear as to what types of instructions and functions are being referred to. Claim 1 recites "... said instructions and data directing said network processor to provide at least one function". Claim 2 recites the "... at least one function is selected from the group consisting of a network emulator, a network profile generator, a network profile capture tool, a packet generation tool, an application traffic generation tool, a real-time packet analysis tool, and a network packet capture and analysis tool." Therefore, the claims state that the instructions are defined as instructions directing the network processor to provide at least one function, and further claim 2 recited that the function is selected from a group of functions. In this amendment claim 2 has been cancelled and the elements of claim 2 added to claim 1, such that claim 1 now states the instructions are used to direct the network processor to provide at least one function and that the functions provided by the network processor include at least one function selected from the group consisting of a network emulator, a network profile generator, a network profile capture tool, a packet generation tool, an application traffic generation

REST AVAILABLE COPY

tool, a real-time packet analysis tool, and a network packet capture and analysis tool. Similar changes have also been made to independent claims 15 and 26. Accordingly, the Examiner's rejection of claims 1-39 under §112, second paragraph is believed to have been rendered moot.

The Examiner has made a provisional double patenting rejection regarding claims 1 - 39 over claims of co-pending application no. 09/920,482. Upon an indication of allowance, Applicants will promptly file a terminal disclaimer.

The Examiner has also made a provisional double patenting rejection regarding claims 1 - 39 over claims of co-pending application no. 09/920,469. Upon an indication of allowance, Applicants will promptly file a terminal disclaimer.

#### Rejections under §102

The Examiner rejected claims 1-32 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,845,352 to Wang et al. (hereinafter Wang).

Wang recites the use of a Finite State Machine (FSM) as part of a simulation engine. At column 5, lines 5-15 Wang states that emulation Manager preferably contains a finite state machine (FSM) that maintains a status of the emulation.

In contrast to Wang, claim 1 recites the use of a network processor. As is known to one of reasonable skill in the art, a network processor is different from a FSM. A network processor is described in the specification as filed at page 6, lines 5-10, which states:

The network processor is typically utilized to perform packet processing, cell processing, look-up table processing and queue management within a network switch or router. The present invention utilizes a network processor in a completely different manner by programming the various processors of the network processor to provide test system functionality instead of switching and routing functionality.

Submitted herewith are three articles supporting Applicants position regarding the differences between a Finite State Machine (as recited by Wang) and a network processor. The articles include a definition of a Finite State Machine, obtained from the Wikipedia website at [http://en.wikipedia.org/wiki/Finite\\_state\\_machine](http://en.wikipedia.org/wiki/Finite_state_machine), a white paper by David Husak titled "Network Processors: A Definition and Comparison" and a presentation by Jacob Engel titled "Network Processor Trends & Design". These references clearly show that a network processor is distinguishable from a Finite State Machine.

By way of claim 1, a network processor, which is conventionally used to provide switching and routing functions in a network switch or router, is used in a different manner to provide test system functionality. Wang fails to disclose or suggest the use of a network processor (instead utilizing a FSM) to perform test system functions.

Therefore, since claim 1 recites using a network processor to perform test system functions, while Wang utilizes a FSM, claim 1 is believed allowable over Wang. Claims 15 and 26 include similar language as claim 1, and are believed allowable over Wang for the same reasons that claim 1 is allowable over Wang. Claims 3-14, 17-25 and 27-42 depend from claim 1, 15 or 26 and are believed allowable as they depend from a base claim which is believed allowable. Accordingly, the rejection of claims 1-39 is believed to have been overcome.

Claims 40-42 have been added. Support for claims 40-42 can be found in the specification as filed at page 6, lines 5-10. Applicants submit that no new matter has been added by the addition of claims 40-42, and further that the addition of claims 40-42 does not require any additional search by the Examiner.

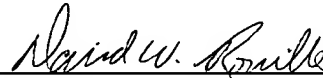
The prior art made of record is not believed to disclose or suggest the present invention.

In view of the above, the Examiner's rejections are believed to have been overcome, placing claims 1, 3-15, 17-26 and 28-42 in condition for allowance, and reconsideration and allowance thereof is respectfully requested.

There is no fee required. If the U.S. Patent and Trademark Office deems a fee necessary, this fee may be charged to the account of the undersigned, Deposit Account No. 50-3735.

If the enclosed papers or fees are considered incomplete, the Patent Office is respectfully requested to contact the undersigned collect at (508) 616-9660, in Westborough, Massachusetts.

Respectfully submitted,



---

David W. Rouille, Esq.  
Attorney for Applicant(s)  
Registration No.: 40,150  
Chapin Intellectual Property Law, L.L.C.  
Westborough Office Park  
1700 West Park Drive  
Westborough, Massachusetts 01581  
Telephone: (508) 616-9660  
Facsimile: (508) 616-9661  
Customer No.: 58406

Attorney Docket No.: EMP04-57

Dated: January 5, 2006

# Finite state machine

From Wikipedia, the free encyclopedia.

A **finite state machine** (FSM) or **finite automaton** is a model of behavior composed of states, transitions and actions. A state stores information about the past, i.e. it reflects the input changes from the system start to the present moment. A transition indicates a state change and is described by a condition that would need to be fulfilled to enable the transition. An action is a description of an activity that is to be performed at a given moment. There are several action types:

Entry action

execute the action when entering the state

Exit action

execute the action when exiting the state

Input action

execute the action dependent on present state and input conditions

Transition action

execute the action when performing a certain transition

FSM can be represented using a state diagram (or state transition diagram) as in figure 1. Besides this, several state transition table types are used. The most common representation is shown below: the combination of current state (B) and condition (Y) shows the next state (C). The complete actions information can be added only using footnotes. An FSM definition including the full actions information is possible using state tables (see also VFSM).

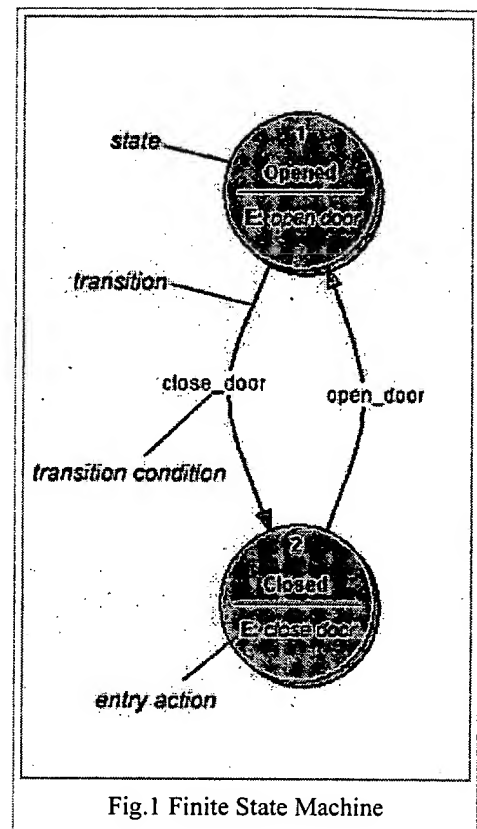


Fig.1 Finite State Machine

State transition table

Current State/ Condition	State A	State B	State C
Condition X	...	...	...
Condition Y	...	State C	...
Condition Z	...	...	...

In addition to their use in modeling reactive systems presented here, finite state automata are significant in many different areas, including linguistics, computer science, philosophy, biology, mathematics, and logic. A complete survey of their applications is impossible here. Finite state machines are one type of the automata studied in automata theory and the theory of computation. In computer science, finite state machines are widely used in modelling of application behaviour, design of hardware digital systems, software engineering, compilers, study of computation and languages.



**By: David Husak**  
C-Port Founder and  
Chief Technical Officer

**White  
Paper**

**Freescale Semiconductor, Inc.**

## Network Processors: A Definition and Comparison

A growing class of communications silicon, the *Network Processor*, promises to revolutionize how networking vendors architect, develop, and support their products. Network Processors deliver dramatic improvements in time-to-market, product lifetime, and system capabilities. This paper examines the benefits of Network Processors in comparison to other networking silicon offerings.

### A Brief History of Network Product Design

The design of networking products has undergone continuous evolution as the speed and functionality of local and wide-area networks have grown. In the early days of packet-based networking, networking devices (such as bridges and routers) were built with a combination of general purpose CPUs, discrete logic, and ASSPs (Application Specific Standard Products), including interface controllers and transceivers. The software-based nature of these devices was key to adapting to new protocol standards and the additional functionality required by networks, such as the early Internet. Although these designs were large, complex, and comparatively slow, they met the needs of these early networks (generally comprised of a few Ethernet or Token Ring connections and slow (56kbps) wide-area links).

Over time, as network interface speeds and densities increased, the performance of general-purpose processors fell short of what was needed. This led network vendors to develop simpler, fixed-function devices (such as Layer 2 Ethernet switches) that could be built with ASICs (Application Specific Integrated Circuits). These devices traded-off the programmability of software-based designs for hardware-based speed. As ASIC technology progressed (and vendors invested heavily in hardware-oriented design teams), more and more functionality was incorporated into the hardware. This was enabled in part by protocol consolidation around IP and Ethernet as the dominant enterprise network technology, which reduced the need for product flexibility.

The relative simplification of network products has allowed merchant silicon vendors to "commoditize" some networking segments through specific chipsets, such as Layer 2 Ethernet "switch-on-a-chip" products. Some of these solutions offer significant functionality within a narrow range of applications, such as ATM switching or basic Ethernet/IP switching. However, network vendors seeking clear product differentiation still required long and risky internal ASIC development programs.

### Today's Network System Development Challenge

*"It's the software, stupid!"*

Vint Cerf, Senior VP for Internet Architecture and Technology MCI WorldCom, and "Father of the Internet" ComSec Seminar, January 1999

Today, the convergence of public voice and data networks is speeding up the pace of change in the communications industry. This is leading to increased time-to-market pressure and shorter product lifecycles — just when product development cycles are growing due to complex ASIC designs and associated software re-designs.

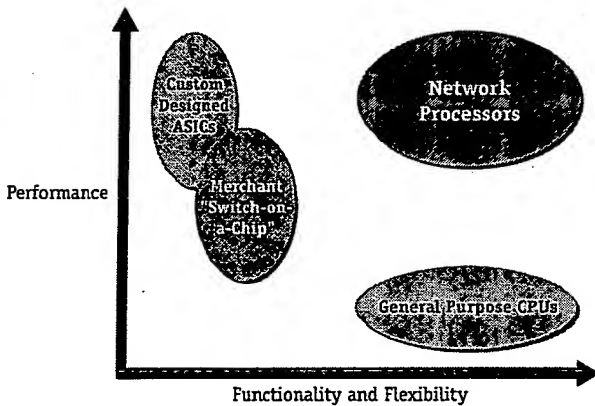
Although IP is emerging as the dominant protocol, newly defined IP capabilities, such as Quality of Service (QoS) and Multiprotocol Label Switching (MPLS), require vendors to continually support new applications. In addition, the number of different interface types,

**For More Information On This Product,  
Go to: [www.freescale.com](http://www.freescale.com)**

ranging from sub-T1 through OC-48 in the WAN space in addition to 10/100 and Gigabit Ethernet in the LAN space, is increasing rather than decreasing.

As a result, networking products require the same programmability and flexibility that was available in the early CPU-based architectures in order to quickly adapt to emerging standards, while maintaining the performance gains achieved through ASICs. To accomplish this, a radically new approach is required. See Figure 1.

**Figure 1** Network Processors Are the New Approach



## Network Processors: Universal Programmability and Performance

Network Processors, emerging on the market today, deliver hardware-level performance to software programmable systems. This powerful combination offers a revolutionary approach to the design of communication systems. It allows systems designers to focus on higher-level services and ensures longer product lifecycles, rather than simply meeting the "speeds and feeds" of the moment.

The power of *true* Network Processors is best examined in light of the seven attributes that are listed in Table 1 and described in the following sections. These attributes are derived from next-generation network requirements for programmability, performance, and openness.

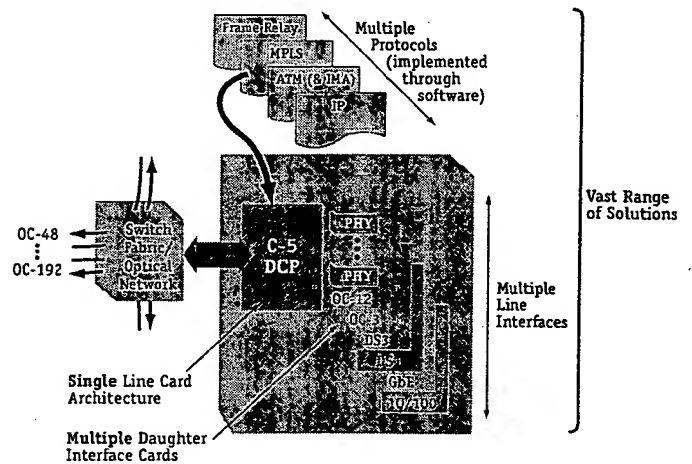
**Table 1** Network Processor's Seven Key Attributes

Attribute	Benefit
Complete programmability	Supports universal networking applications
A simple programming model	Leads to faster time-to-market
Maximum system flexibility	Enables longer time-in-market™
Massive processing power	Provides scalable performance
High functional integration	Lowens total system costs
Open programming interfaces	Delivers higher availability
Third-party support	Encourages continuous innovation in the industry

## Complete Programmability

For real platform leverage, a Network Processor must be universally applicable across a wide range of interfaces, protocols, and product types. This requires programmability at all levels of the protocol stack, from Layer 2 through Layer 7. Protocol support must include packets, cells, and data streams (separately or in combination) across various interfaces to meet the requirements of carrier edge devices, for example, that are the cornerstone of the emerging multiservice carrier network. See Figure 2.

**Figure 2** Universal Switch-Router Line Cards Based on Network Processor



This type of multiprotocol solution offers important time-to-market competitive advantages, and dramatically reduces support costs for both the network vendor and service provider.

### Simple Programming Model

The programmability of a Network Processor must be readily accessible to the developer in order to be useful. By far the most common software languages in real-time communications systems are C and C++, with millions of skilled programmers and many more lines of existing code.

Programming in the C and C++ languages also enhances the future portability of the code-base, enabling use in future generations of Network Processors and industry standard programming interfaces. This is not possible with specialized languages or state-machine codes.

### Maximum System Flexibility

True Network Processors integrate all the functions implemented between the physical interfaces and the switching fabric, enabling an open approach for the PHY and fabric levels. This permits best-of-breed, multi-vendor solutions that allow vendors to offer true product differentiation and scalability. In addition, software implementation of these functions allows simpler upgrade paths in this constantly changing networking world.

### Massive Processing Power

The architecture of the Network Processor needs to be more than the amalgamation of a few RISC core processors and some packet processing state machines. A fully optimized processing architecture, with a high MIPS (millions of instructions per second) to Gbps (Gigabits per second) ratio is required to support wire-speed operation at high bandwidths and still have processing headroom for advanced applications.

### High Functional Integration

Network Processors need to provide a high level of system integration that dramatically reduces part count and system complexity, while simultaneously improving performance, as compared to using a design that incorporates multiple components (such as ASSPs).

In addition, a highly integrated Network Processor avoids the interconnection bottlenecks common with component oriented designs. Integrated coprocessor engines (such as for classification or queuing) can be fully utilized by internal processing units without interconnection penalties.

Integration of lower layer functions (such as SONET framers) within the chip also enables higher port densities and lower costs than have typically been possible in the past.

Figure 3 and Figure 4 provide a comparison of a multiple component system versus a highly-integrated system.

Figure 3 Typical Interworking Design Using ASSPs and CPU

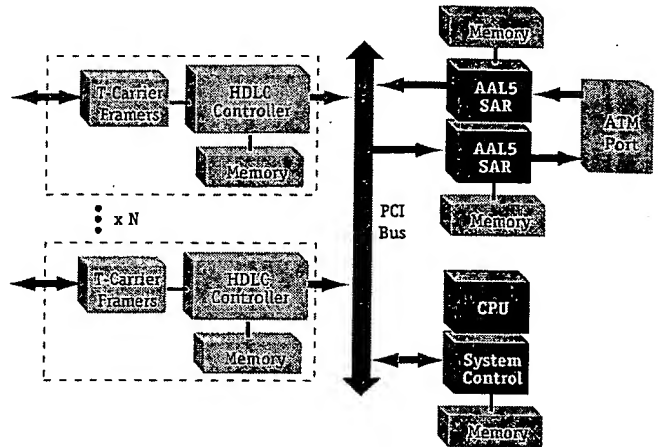
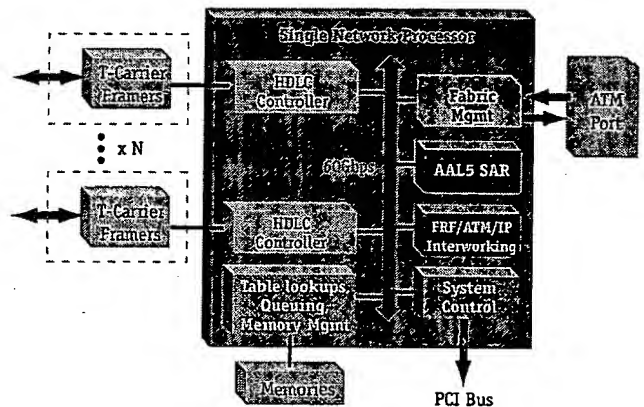


Figure 4 Interworking Design Using a Highly Integrated Network Processor



### Stable Programming Interfaces

A communication processor cannot deliver on software flexibility and portability if the programming interfaces are dependent on the processor. The processor's architecture must support generic "Communications Programming Interfaces" to simplify the programming task and allow future software reuse across generations of the processor.

By delivering software stability across product generations, Network Processors radically improve software development cycles and reliability. Software reliability is the largest factor in total system availability.

### Third-Party Support

To realize the full potential of a software-driven environment, the Network Processor needs to be the foundation of a complete communications platform that takes advantage of industry-wide hardware extensions,



software applications, and tool suites. This is only possible with an architecture that has the flexibility to support virtually any third-party protocol stack, any PHY or fabric interface, and links with industry standard tools. Such broad support significantly decreases time-to-market.

## Network System Design Alternatives

Of course, before the Network Processor there were a number of design alternatives that in their own ways provided some assistance in building better networking products. From using completely hard-wired solutions to configurable processors, and more recently, network processor chipsets, networking vendors have incrementally improved and evolved their designs, but not without major compromises.

### Custom ASIC Designs

Until recently, common practice of high-speed networking design has involved the development of custom ASICs for critical elements of the architecture. This approach has been dictated by the requirement for "wire-speed" performance at reasonable cost.

Most vendors have had limited success in leveraging an ASIC or ASIC family into multiple product lines, preventing them from amortizing the development costs across a broad range of revenue-generating products. Implementing product architectures in ASICs is a high-cost proposition from a number of perspectives:

- The design cycle is typically 18 months (and can extend beyond 3 years). Projecting market requirements that far in advance is difficult given the competitive dynamics of the market, resulting in the same company needing to place "multiple bets" to assure market success.
- The risk of design failures in ASIC-based development is large, given the many months that are often required to correct design flaws (due to the lack of flexibility present in hardware-based designs).
- The limited flexibility of hardware-based designs severely limits the ability to adjust product functionality to evolving market demands before and after market introduction. The result is shorter product lifecycles and greater to-market risks.
- ASIC design expertise is a rare commodity. The ability to hire and retain talented designers has become a fundamental limit to the rate of product development for many vendors.

- The design tools for complex ASICs can run in the millions of dollars (ASIC emulators, for instance) and require constant refresh as the technology advances.
- Perhaps the largest, and often hidden cost, is the need to re-architect and re-write critical software associated with each product generation. Frequently, the extent of the re-write is unforeseen and prescribed by the need to optimize the designs around the hardware and ASIC technology, rather than around software re-use. Thus regardless of how much key functionality is embedded in the hardware, a massive amount of "slow-path" software is generally still required.

Additionally, the large opportunity cost of software re-writes often prevents vendors from delivering the value-added services and applications that provide true market differentiation.

While there remains a set of products that will require the customization available from ASICs, most vendors are eager to move to design alternatives that will improve their time-to-market and reduce development risks.

### Customizable ASICs and Configurable Processor Designs

Several new design technologies are emerging to address some of the issues and risks of ASIC-based designs. These include:

- Integrated Circuits (ICs) incorporating fixed-function network logic blocks with configurable interconnects (sometimes termed "systems-on-a-chip")
- Configurable processor cores with changeable instruction sets that allow limited modifications to accomplish some network-specific tasks

#### Configurable "Systems-on-a-Chip"

Configurable "system-on-a-chip" approaches mix a number of fixed-function blocks, perhaps including a CPU-core, on a single chip with FPGA-like configurable interconnects. These devices speed-up the development cycle by enabling designers to choose from a "menu" of available functions that they assemble to build the desired part. Such devices sometimes promise future field re-configurability of the interconnection between different elements.

While this approach offers some time-to-market advantages compared to traditional ASICs, having a collection of fixed-function blocks limits the flexibility to adapt to new features and standards because the design

remains essentially in hardware. In addition, having a single software-capable CPU element limits both performance and programmability.

### **Configurable Processor Cores**

Configurable processor cores embedded in an ASIC design allow customization of the instruction set. In networking applications, this may allow the designer to create specialized instructions for certain communications tasks, such as the implementation of specific software encryption algorithms. However, such architectures assume that the software is handling the data and thus the processor must be inserted in the primary data path. This approach does not scale and has been the main reason discrete CPU-oriented systems have failed to keep pace in the past. Moreover, this approach also does not address the fundamental bottleneck in "soft" architectures — separating the control path from the data path in an effective and scalable way.

Both the configurable "systems-on-a-chip" and configurable processor cores are variations on the custom chip theme, and suffer in varying degrees the limitations described above for ASIC-based approaches. In particular, these methods do not address maximizing software re-use as the key to higher reliability, faster time-to-market, increased product lifespan, and the delivery of expanded network services.

### **Application-Specific Standard Products**

Most communications system designs, whether based on CPU or ASIC architectures, make use of some specialized components. These are often used where a specific function is difficult to build into an ASIC, is available in a low cost off-the-shelf component, or is not central to the system design. An example of an ASSP-based design is shown in Figure 3. Some silicon vendors have continued to make standalone ASSPs attractive by supplying increasing functional density, such as has occurred with low-speed framers and physical interfaces (transceivers).

### **Smart MACs**

Communications IC vendors have also been working to make some components smarter, integrating more and more of the functions that would normally be handled by a CPU and software (or in hardware with a custom ASIC) into the component.

For example, some makers of Ethernet MAC ICs have begun incorporating some protocol data parsing and processing functions within the interface, alleviating some of the

lower-level tasks from the system software. This can be beneficial for certain classes of products (such as network interface cards, for example), but is really only a small incremental improvement over traditional design methodologies for mainstream communication systems.

### **Single-Function Components**

A different approach has been taken by other vendors who have set out to design optimized components addressing a single, higher-level function within the system. Examples include IP address lookup engines and traffic classifiers. These components reduce the number of functions the system designer must implement in custom ASICs and subsequently reduce time-to-market. Some of these components also represent the state-of-art for a particular function, increasing the capability of the system solution.

However, systems based on these devices still suffer from the limitations of a hardware-oriented approach. The configurability provided in the components is usually only enough to support the specific design, but not enough to adapt to emerging customer requirements or standards. Higher-level services, which must be implemented in software, are also limited (if not prevented) by this approach.

Perhaps the biggest obstacle to single-function components is the level of effort required to effectively integrate various components into a complete system. For the higher bandwidth interfaces (like Gigabit Ethernet, OC-12, and OC-48), the interconnect design between components is often the primary system bottleneck. Multiple components lead to more complex hardware designs, less scalability, and increased time-to-market.

### **Programmable Communications Components**

There are classes of communications-focused ICs with programmability similar to Network Processors. No matter how programmable a specific component may be, however, it is still limited by the overall system design issues of integrating various independent components. See Figure 5.

### **Digital Signal Processors**

Digital Signal Processors (DSPs) offer a great deal of flexibility in the implementation of signal processing algorithms for a wide range of physical layer applications such as high-speed modems. With custom instruction sets for fixed and floating point arithmetic, DSPs are optimized for the mathematically intensive algorithms used in advanced signal processing.

Some vendors have proposed using DSPs (or multiple DSP cores on a single chip) to expand beyond pure signal processing into higher-level protocol handling. While some protocol processing may be supported within DSP architectures, the basic tasks of data formatting, parsing, classification, modification, and switching are fundamentally different from the mathematically oriented tasks of DSPs.

The tools for programming DSPs are also oriented toward algorithmic implementations and require specialized language support. So, while DSPs are a great example of the power of programmability within communications systems, they are not an adequate universal processing solution for the higher-level protocol processing functions.

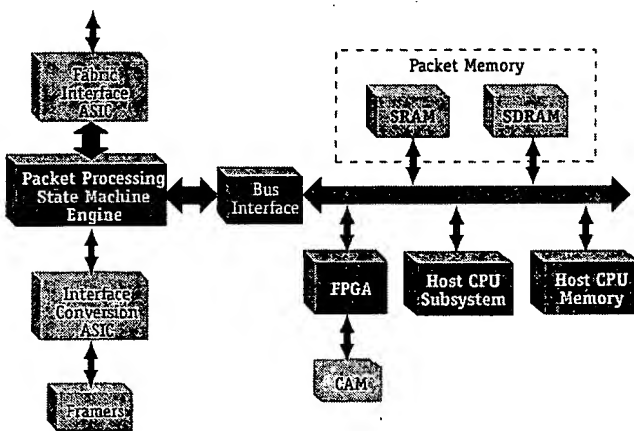
#### Configurable State Machine Engines

Another approach for achieving flexibility at the component level is the application of configurable state machine engines for off-loading some of the protocol processing from general purpose CPUs.

These devices have sometimes been classified as "network processors," although they do not execute any "software" in the traditional sense. Instead, they have a series of configurable state machines that perform some of the framing, data parsing, and classification functions. Based on the configuration, these devices may pre-process ATM, Frame Relay, or Ethernet formatted data for a general-purpose CPU, for additional components (such as a MAC, classification engine, or custom ASIC), or for both.

Figure 5 shows a product design using a configurable state machine engine.

**Figure 5** Typical State Machine Engine-Based Design



The state machines are configured through CPU-accessible registers or external devices (such as FPGAs). Because the configuration of state machines can be quite complex, some vendors implement the required functions using specialized procedural languages to generate the actual state machine code; while other vendors provide a suite of pre-configured codes for a variety of 'canned' applications.

Although these state-machine-oriented devices offer more flexibility than typical fixed-function ASSPs, they suffer from the same architectural limitations. The design must still revolve around a general-purpose CPU or a custom ASIC in the switching path, with the requisite performance, flexibility, and time-to-market trade-offs.

#### Programmable Special Purpose Devices

Some communication component vendors have focused on increasing the programmability of their single-function components in order to provide better future adaptability and to broaden the market appeal of their devices.

An example is segmentation and reassembly (SAR) devices, designed specifically to perform the interworking between frame-based (Ethernet and IP) networks and ATM-based networks. SAR architectures typically consist of Utopia interfaces, frame and cell parsing logic, dedicated scheduling and queuing support, and a custom processor for implementing the interworking protocols. A software-oriented processor is attractive in SAR components due to the rapidly evolving ATM interworking standards.

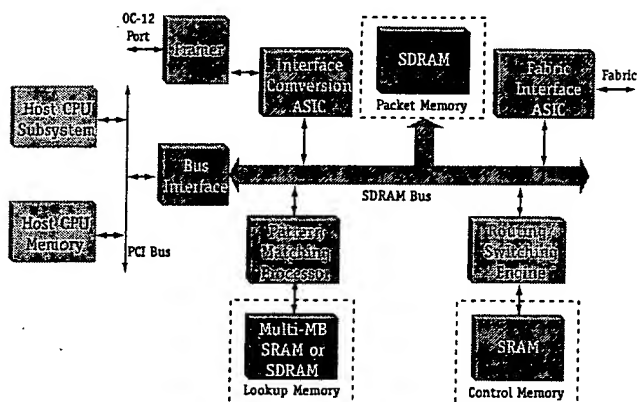
Vendors of other components, such as HDLC controllers, are also allowing the "extra" processing cycles within their devices to be used for customer-defined applications.

There are many difficulties when applying these devices beyond their originally intended purpose (SARing, HDLC multiplexing, and so on). For example, it can be difficult to determine exactly how many "extra" cycles are really available for custom processing. Further, the internal processors themselves are typically proprietary CPUs, specifically designed for one function. This means questionable suitability to more general processing tasks, often surprisingly large impacts on system performance, and the possible immaturity of the programming tools.

### Pattern Matching Processors

Another approach at providing programmability is a further extension of the single-function component. Examples of this include "pattern matching processors" that focus on providing configurable classification engines (see Figure 6).

**Figure 6** Typical Pattern Processor Design (OC-12 WAN Interface)



Pattern matching processors provide more flexibility and configurability than the fixed-function devices described above, even allowing support for multiple protocol types (ATM, IP, and so on). The value of these devices is in embedded algorithms specifically useful for classification, which are sometimes configurable through a proprietary programming language.

Aside from the obvious issues with proprietary languages, it is often difficult to evaluate the performance of these processors within an overall system design, due to the performance links between the classification functions and the switching and routing functions that must be implemented elsewhere in the design.

### Network Processor Chip Sets

One of the fastest growing areas of merchant communications silicon is in the area of switching chipsets. Many ATM switching platforms are based on standard silicon, as are most low-end Ethernet workgroup switches.

For the low-end systems, the obvious benefits are the ability to develop commodity-oriented products quickly and at very low cost. While at the high-end, systems are often built with a mixture of the standard components that make up the chipset and custom designs (usually ASICs) that provide vendor differentiation.

Among the high-end switching chipsets are the early "network processors", offering a complete fabric and packet processing solution for systems ranging up to multi-gigabit performance. Some of these architectures support both packet and cell-oriented systems, though not at the same time. Additionally, there may be fixed, limited interfaces within the architecture that enable networking vendors to program a small amount of functionality, or pass data to an external device (usually of custom design).

A key drawback of the "switch-on-a-chip" and network processor designs is the limited flexibility for providing system differentiation or additional services beyond those envisioned by the original silicon architects. Invariably, the instruction sets provided in the architecture are proprietary, primitive, and have limited tool support. It is not uncommon for designers to discover the performance and functional limitations of these interfaces late in the design cycle, forcing time-to-market delays and critical functional trade-offs.

Most of these solutions also use proprietary interconnects between the various chipset components, from port processing through switching fabric. Not only does this limit the ability of the networking vendor to choose "best-in-class" solutions, but the silicon architecture tends to "blur the lines" between functions. The end result is limited scalability of the final product, preventing future growth and adaptability of the product line. Such an "all or nothing" approach to system design can often be difficult for networking vendors to accept for strategic product lines.

For commodity-oriented communication products, complete "switch-on-chip" solutions can be viable time-to-market approaches. However, for higher-end products that must live in a complex and evolving application environment, an open approach (from both hardware and software perspectives) is required.

**Table 2** Comparison of Network System Design Approaches

	Complete Programmability	Simple Programming Model	Maximum System Flexibility	Massive Processing Power	High Functional Integration	Stable Programming Interfaces	Third-Party Support
Network Processors	++	++	++	++	++	++	++
Custom ASICs	--	--	--	++	+	--	--
Configurable Processors							
Configurable SOC	+	--	+	-	+	--	-
Configurable Processor Cores	+	-	+	-	+	--	--
Application-Specific Standard Products (ASSPs)							
Smart MACs	-	-	-	--	-	--	--
Single Function Components	-	--	-	+	--	-	--
Programmable Communications Components							
DSPs	+	-	+	+	-	+	+
State Machine Engines	+	--	-	-	-	--	--
Special Purpose Devices	+	-	+	-	--	-	-
Pattern Matching Processors	+	-	-	+	--	--	--
Switching Chipsets							
L2 chipsets	--	-	--	--	++	--	--
Network Processor chip sets	+	--	-	-	+	--	-

++ is excellent; + is good; - is fair; and -- is poor

## Summary

As you can see, Network Processors offer a revolutionary way of developing networking products that deliver dramatic improvements in time-to-market and time-in-market™. Table 2 provides a comparison of Network Processors to the alternatives discussed in this paper.

Only true Network Processors, like the C-Port C-5 Digital Network Processor (DCP), offer all of these significant benefits:

- **Complete programmability** — At all protocol layers from Layer 2 through 7, enabling adaptability to a wide range of requirements at any point in the network hierarchy.
- **Simple programming model** — Leveraging well known programming methods and languages (C and C++) to allow faster time-to-market and portability of code across platforms.
- **Maximum system flexibility** — Maintaining a “soft” approach to enable new services and standards to be deployed with software-only upgrades.
- **Massive processing power** — Required to fully and robustly implement the key networking functions and new services, and deliver wire-speed operation at high bandwidths.

- **High functional integration** — Implementing all the network functions in a single chip solution to lower total system costs.
- **Stable programming interfaces** — Simultaneously simplifying programming tasks and maximizing software reuse for future product generations.
- **Third-Party support** — Leveraging the proven solutions of industry leaders in the software and hardware development community for faster time-to-market and better reliability.

Because of these advantages, networking vendors can leverage the power and flexibility of software to apply a platform approach to system development, and focus more R&D resources on delivering the functions and services demanded in today's highly competitive market.



**C-PORT.**  
A Motorola Company

C-Port Corporation  
One High Street  
North Andover, MA 01845  
978-773-2300 TEL  
978-773-2301 FAX

www.cportcorp.com  
www.mot-sps.com

C-5, C-Port, C-Ware CPI, C-Ware Partner, the C-Port logo, and time-in-market are all trademarks of C-Port Corporation.

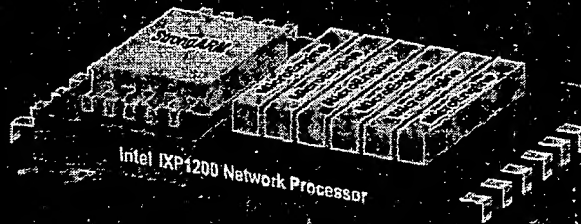
© 1999, 2000 C-Port Corporation  
CP00WP201

**For More Information On This Product,  
Go to: [www.freescale.com](http://www.freescale.com)**

# Network Processor

Trends & Design

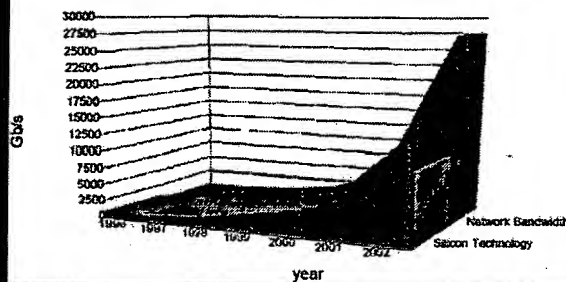
Jacob Engel



FILE COPY

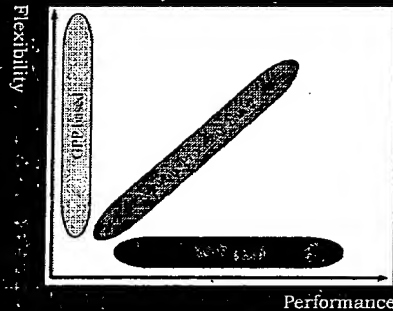
## Network Trends: Bandwidth

Projected Network Demand (Gb/s) vs Moore's Law



- Increasing network traffic
- New sophisticated protocols are introduced at rapid pace
- Voice & Data convergence

## NP evolution & functionality



### General Purpose Processor (GPP)

- Advantages
  - Quick time-to-market
  - Core performs all routing functionalities
  - Flexible to upgrade the system
- Disadvantages
  - Not easy to scale up the system
  - Major reduction in performance for complex operations (traffic management, QoS)

### NP-based

- Advantages
  - Provides high performance & flexibility
  - Supports multiple threads (multiple PEs)
  - Uses pipelining + parallel processing
- Disadvantages
  - Not easy to scale up the system
  - Major reduction in performance for complex operations (traffic management, QoS)

### ASIC based

- Advantages
  - Wire-speed performance
  - Integrated with additional GPP, RISC or DSP processors to separate control vs. data
- Disadvantages
  - Lacks flexibility
  - Long design cycle => high time-to-market
  - Change in design/failure => high risk
  - Complex operations still done in SW

## Network Processor Evolution

*A network processor is a programmable device that has been designed and highly optimized to perform networking functions.*

### Generation I

#### RISC-based

- Numerous instructions => more time
- Trying parallel => increasing sys complexity and chip size

### Generation II

#### Augmented RISC-based

- RISC + ASIC => accelerate hardware
- Less flexible

### Generation III

#### Network specific processor

- Many small & fast processor cores

## Key characteristics of the architecture

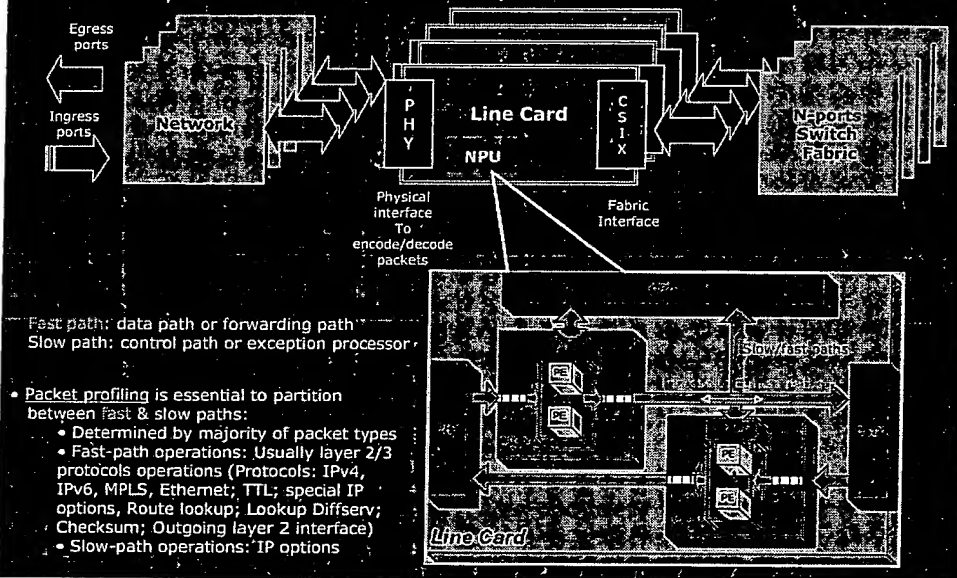
- **Programmable**  
The essence of a NP is that it is programmable. High performance is maintained by implementing the NPs with their own optimized instruction set for the task of processing network packets.
- **Modular**  
The architectures employ different schemes and levels of modularity. To obtain high performance and scalability - components of the NP and connecting technology must be programmable to perform the work they are best suited for.
- **Scalable**  
Most of the architectures discussed allow network vendors to build small, inexpensive to large, high-performance network devices. NP vendors have some kind of switching fabric that takes the place of the central bus and offers a full crossbar switching architecture.
- **Parallel**  
Many of the architectures have micro-engines that can be programmed and can intelligently process data at wire speeds. They accomplish high throughput by supporting multiple thread on each engine (Multithreading). Other designs employ super pipeline and superscalar architectures for massive processing power.

## Key characteristics of the architecture

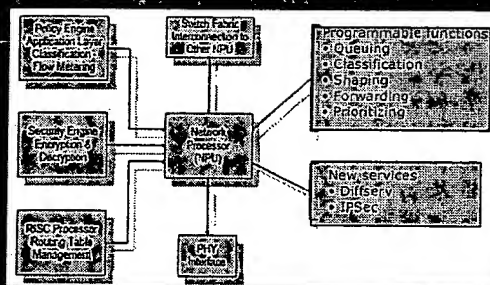
- **Integrated bus**  
Typical NP architecture has some type of integrated bus. This bus integrates the processor cores, the memory systems, interfaces to the physical adapters and the host system bus. This integration reduces part count and system complexity, while improving performance.
- **High-bandwidth memory**  
High-bandwidth, low latency memory is essential for the network processors to achieve the necessary speed required. The architectures either embeds the needed memory on the chip or employs scratchpad memory, high register counts and a sophisticated interface to external memory than reduces contention.
- **Wire-speed intelligent processing**  
To support features such as QoS algorithms are being developed to classify, mark and regulate packet flow. The hardware offers the opportunity to perform these operations at wire speeds with well-designed and well-implemented algorithms.
- **Table lookup algorithms**  
Some designs have special table lookup units that can handle multiple lookup algorithms simultaneously at very high speeds.



## Router Architecture



## TCP/IP packet header & services

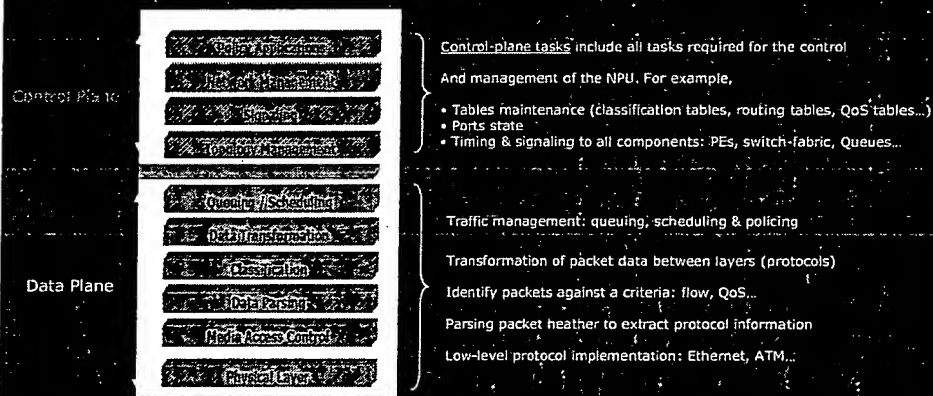


MAG IP TCP APP DATA MAG

Ver.	D/L	Service Type	Total Length
Identification		Options and Fragment Offset	
Time to Live	Protocol	Header Checksum	
Source Address			
Destination Address			
Options (0 or more 32-bit words)			

Source Port	Destination Port
Sequence Number	
Acknowledgement Number	
Header Length and Options	Window Size
Checksum	Urgent Pointer
Options (0 or more 32-bit words)	

# Application categorization

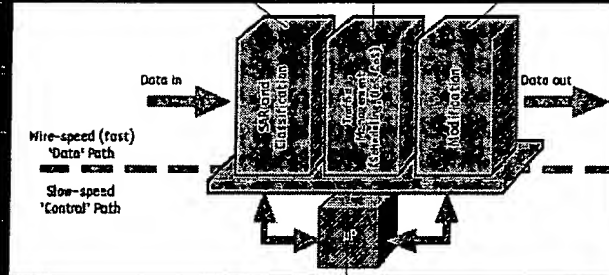


## Data-plane operations - examples

- **Priority based QoS mechanism**
  - Supports different levels of QoS for each output port
  - Contains QoS policy table – prioritizing packets
  - *Ingress operations*
    - Applies QoS policy on the packet received
    - Gets the packet priority from its header content
    - Place the packet in the appropriate output queue
  - *Egress operations*
    - Identifies & schedules highest priority packet for transmission
    - Transmits the identified packet on to the output port
- **Security**
  - Encryption/Decryption, intrusion detection, access control checking, denial-of-service
- **Monitoring**
  - Capturing usage patterns, time information
- **Load Balancing**
  - Distribution of traffic among servers according to the server load, content and client credentials

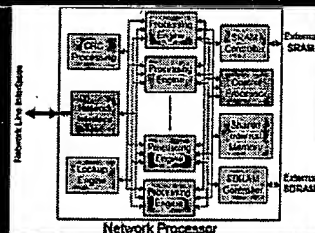
## NP basic architecture & packet processing flow

- Controls data flows across network to optimize network resources providing bandwidth & delay guarantees
- Packet Segmentation into fields
- Classification - inspection of packet properties in order to determine the best way to process it
- Packet header modification needed to process and route to destination



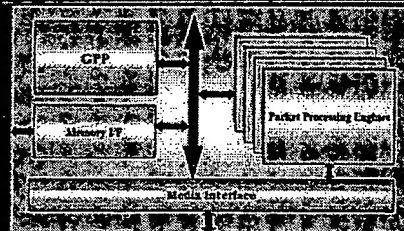
- The uP handles initial setup and exception processing

## High-level Architecture

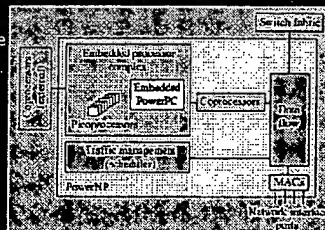


Processor: application management  
Coprocessors: hardware assist (table search, Packet alterations)

External CAM



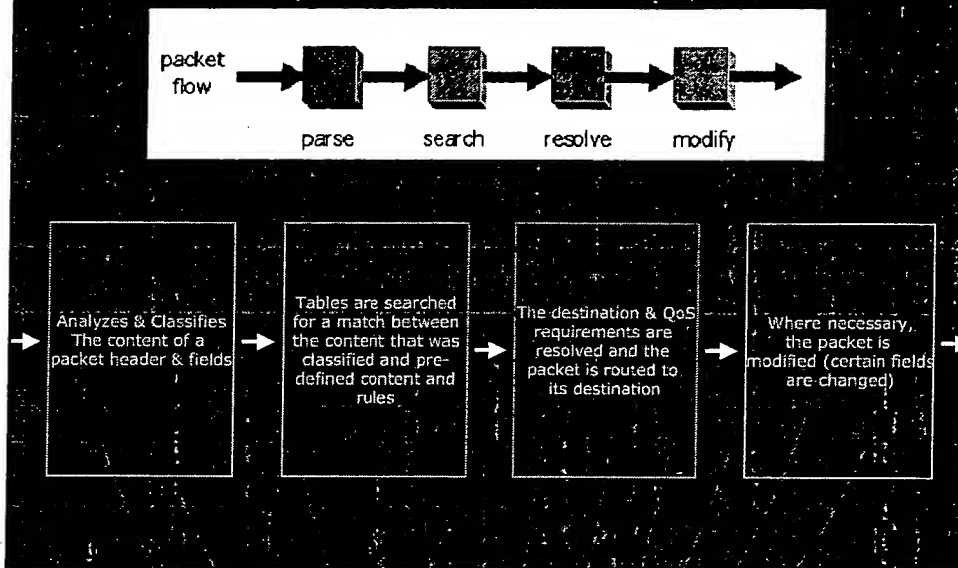
Data flow: Primary traffic data path Provides interface to data memories for buffering



Scheduler: Allows traffic flows to be scheduled individually per their QoS class for differentiate services

Media interface: Ethernet, SONET, ATM

## Network Processor Functions



## 7-layer processing WHY?

Differentiation of networking equipment enables the vendor to provide better service to the customer supporting applications such as:

- Policy-based networking

Enabling fine-grained service provisioning in accordance with flexible policy rules  
Including bandwidth allocation, priority definition, security enforcement, route selection

- Server load balancing

Traffic distribution among servers in accordance with web destination  
URL, client credentials (cookies), server load

Example

NAT (Network Address Translation): layer 3 – IP addresses, layer 4 – port numbers, layers 5-7 – address instantiations

Forwarding frames: layer 2 – MAC addresses, layer 3 – IP addresses

Target server selection: layer 5-7 – URL and cookie extraction

Server and client message handling: layer 5-7 cookie modification

## 7-layer processing WHY?

- Usage-based accounting

Provision of detailed per-flow, network bandwidth usage for billing & capacity planning

Example tracking the download and bandwidth usage to specific client when downloading a video clip

Recognize session initiation for specific server – layer 3 IP address and layer 4 port number

Monitor login session to identify the user name – layer 5-7 extraction of login info

Identify desired file name to download – layer 5-7 extraction of file name and matching to program policy tables

- MPLS traffic engineering

Delivery of highly-granular QoS required for time sensitive applications  
Voice & video over IP; real-time business transactions

- Network monitoring & analysis

Intelligent network management, alert notification and troubleshooting

## The memory challenge

### The challenge

- As bandwidth requirement grow => network speed increase => memory becomes bottleneck
- 7-layer processing (deep packet processing) places heavy strain on memory usage (reading and writing significantly more data to / from memory)
- Packets storage and lookup table searches

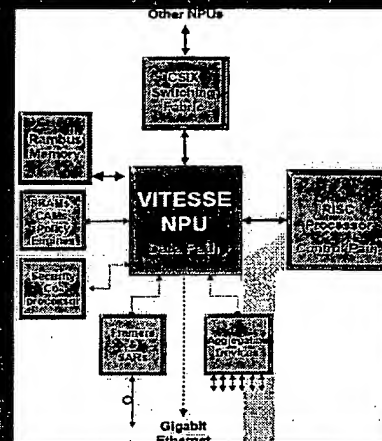
Packet buffer memory => DDR DRAM / SRAM / RAMBUS  
This buffer memory is accessed at least 4 times per packet

- Writing the packet when received from the network
- Reading frame data for processing (lookup tables)
- Writing modified packet for transmission
- Reading the packet for transmission

Therefore, in order to sustain wire-speed performance the PBM should provide at least 4 times bandwidth of the network link. i.e., 40 Gbps

### Lookup tables memory => CAM

During classification headers and data fields of an incoming packet are used for searching in various lookup tables containing QoS, control, policy information, IP addresses  
Therefore, it requires simultaneously table access  
To separate memory cores. When considering Full duplex data transfer we get 160 Gbps memory Bandwidth!!!!

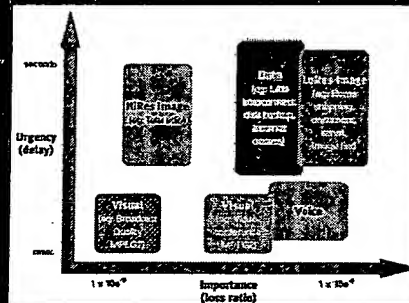


# Quality of service (QoS)

"A mechanism for networks to satisfy the varied quality and grade of service required by application while at the same time maximizing bandwidth utilization"

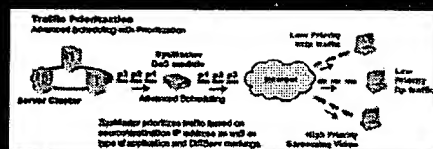
- Applications are classified into groups and these groups are carried across the network over a finite set of QoS classes
- Applications such as interactive video and voice are delay and loss sensitive => supported by constant bit rate (CBR) service
- Applications NOT delay/loss sensitive => variable bit rate (VBR) service
- Example of QoS classes

- ATM classes – six QoS classes offering loss, delay, and jitter guarantees
- IP DiffServ – uses the ToS IP field to differentiate packets. Three main classes: Expedited Forwarding (EF) – low loss/delay ; Assured Forwarding (AF) not as stringent as EF ; Best effort packet switching
- MPLS labels – packets are encapsulated and assigned labels which provide service differentiation

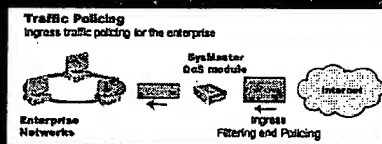


Applications loss vs. delay requirements

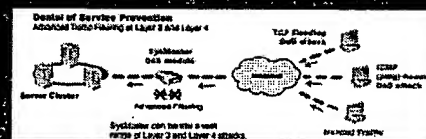
## QoS mechanisms



Traffic prioritization



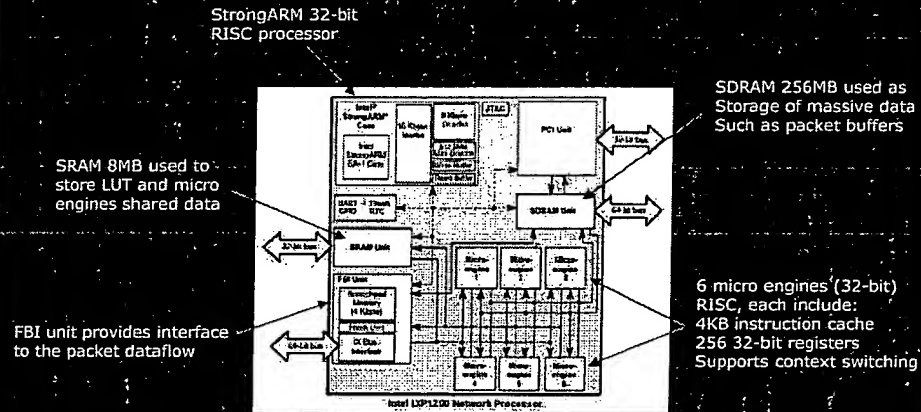
Traffic policing / shaping



Denial of Service (malicious traffic)

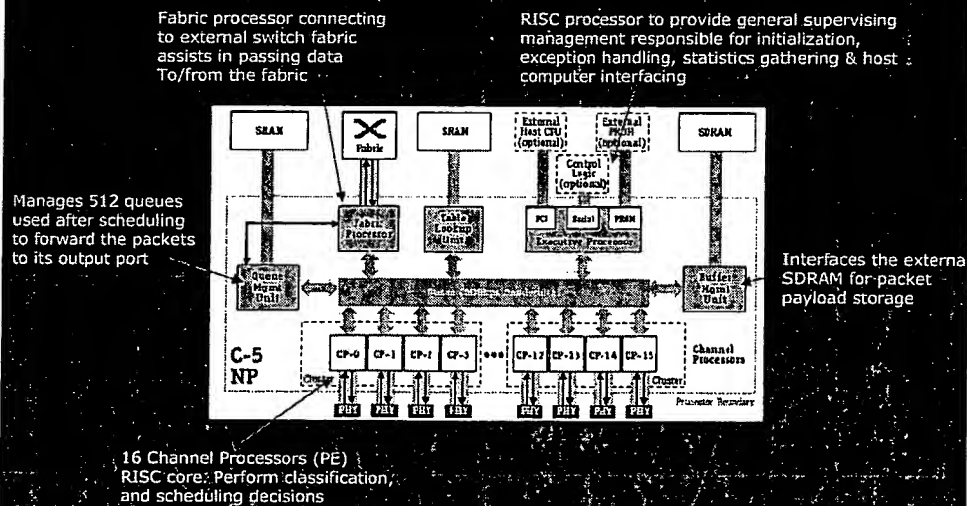
# NP architectural approaches

## INTEL IXP1200



# NP architectural approaches

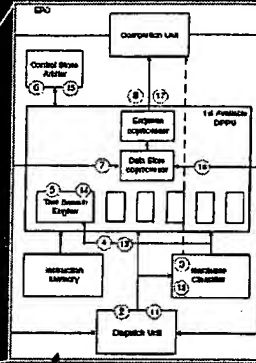
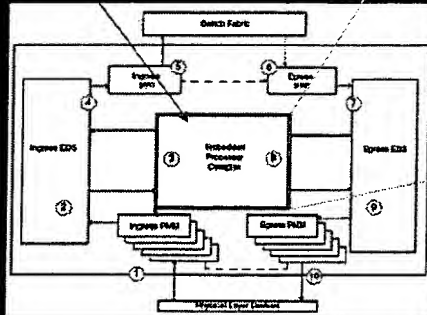
## MOTOROLA C-5



# NP architectural approaches

## IBM Power NP4GS3

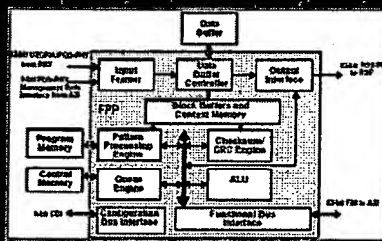
EPC contains 8-multithreaded protocol processing units  
Manages memory allocation among threads  
Dispatches threads Keeps same flow packets in order  
Policy management Tree search engine (internal)



Contains 8 multi-threaded Protocol processing units. Optimized instruction code. Hardware accelerators such as tree searching, CRC, data Stores, enqueueing, policy control.

# NP architectural approaches

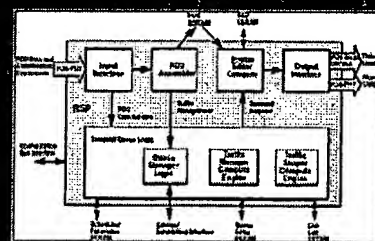
## Agere PayloadPlus



Fast Path Processor (FPP)

Pipelined Multithreaded proc.  
Simultaneous 64 packet classification

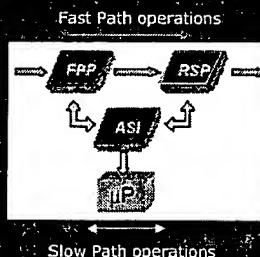
Output result sent to RSP



Routing Switch Processor (RSP)

VLIW processors that can run  
Multiple programs simultaneously

Performs queuing, traffic  
Management, shaping and packet  
modifications

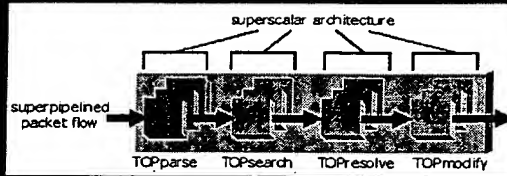




# NP architectural approaches

## EZchip NP-2

10 Gb wire speed 7-layer NP  
Eliminates the need for CAMs  
Supports deep-packet processing: layers 2-7  
Highly programmable to achieve fast time to market



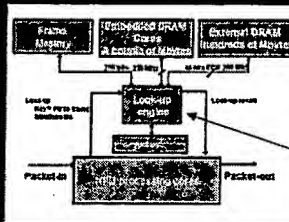
Packet Fields, Tags, Addresses, Protocols, patterns

Lookup Classification policies

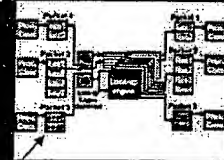
Forwarding Decisions, QoS, updates

Packet Modifications

Protocol independent search engines  
Multiple search engines operating in parallel with pipelining of memory cycles and intelligent partitioning to overcome the DRAM latency barrier



Lookup engine with minimal instruction set recombines data into keys  
Performs chained lookups (a key from one packet can be used to Search another packet)



**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER: \_\_\_\_\_**

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**